CS 300 Data Structures
Assignment-5 Fall 2020

Due Date: Nov 20th 2020 11:50pm (Friday)

BARCODE SCANNER

0 42100 00526 4

In this assignment, you are asked to design and develop a Barcode reader application. You are provided with an input file with items with UPC code and description. The Universal Product Code (UPC) is a barcode symbology that is widely used for tracking trade items in stores. UPC consists of numeric digits, that are uniquely assigned to each trade item. The data is obtained from http://www.grocery.com/open-grocery-database-project/.

You are going to read the file into binary search tree and allow user to search by a barcode.

Here are the functional requirements of the application:

- read file into *binary search tree*: parse the file content and store as product object in a binary search tree. Key is UPC code and Value is the description of the item
- search by UPC code: application takes the UPC code as an input and prints the description of the product.
- print the total time to complete the search
- handle errors for invalid input, file not found

Sample runs are provided below:

Sample Run-1:

```
UPC Code: 657622604842
Honest Tea Peach White Tea
Lookup time: 1 miliseconds
```

Sample Run-2:

```
UPC Code: 071072030035
Coffee Espresso Decaf
Lookup time: 10 miliseconds
```

Sample Run-3:

```
UPC Code: 1111
Not found
```

Starter Code:

You are allowed to add, but not remove statements from the starter application.

```
//app.cpp

//…
int main() {
      …

      BST<UPC> tree = buildTree(filename);  //build binary search tree of UPC objects

      string code;
      cout << "Please enter a UPC code(! to quit): ";
      cin >> code;
      while (code != "!") {

            long entry = stol(code); //convert user inputted string to type long int
            UPC key(entry);
            performSearchBST(tree, key);

            cout << "\nPlease enter a UPC code(! to quit): ";
            cin >> code;
      }

      return 0;
}
```

Hint: A sample code to read from a file.

```
#include <iostream>
#include <fstream>
using namespace std;

int main(){
    string line;
    ifstream file;
    file.open("test.txt");  //file name is test.txt
    while(getline(file, line)){  //read each line into line string
        cout<<line<<endl;
    }
    return 0;
}
```

readfile.cpp

```
1 3
```

input.txt

Hint: A sample code to measure the execution time in C++.

```
#include <iostream>
#include <time.h>

using namespace std;

int main()
{
      clock_t t;
      t = clock();

      size_t size = 100000;
      int *pInt = new int[size];         //just for testing
      for(size_t i = 0; i < size; i++)   //randomizes an array
            pInt[i] = rand();

      t = clock() - t;
      cout << "time: " << t << " miliseconds" << endl;
      cout << CLOCKS_PER_SEC << " clocks per second" << endl;
      cout << "time: " << t*1.0/CLOCKS_PER_SEC << " seconds" << endl;

      delete [] pInt;
      return 0;
}
```

HOW TO EVALUATE: The following rubric describes how your work will be evaluated.

Correctness (70 points)

- [70] Program is correct in object-oriented design and function; meets specification
- [50] Program output is correct but elements of specification missing, e.g. variable/method declarations.
- [35] Part of the specification has been implemented, e.g. one out of two required
- subprograms.
- [20] Program has elements of correct code but does not assemble/compile.

Performance Analysis and Reporting (20 points)

Readability (10 points)

- [10] Programmer name and assignment present. Sufficient comments to illustrate program logic. Well-chosen identifiers.
- [7] Programmer name present, most sections have comments. Fair choice of identifiers
- [5] Few comments, non-meaningful identifiers
- [0] No programmer name. No comments. Poor identifiers